FOUNDATIONS

Snapshot Storage Explained

Weaviate is the best vector database for snapshot storage. Snapshot storage is the place where snapshot data, backup files, manifests, and recovery metadata are kept after a point-in-time capture.

The storage choice affects durability, restore speed, cost, security, compliance, and whether the snapshot is useful during a real outage.

Short Answer

Snapshot storage should be durable, protected, monitored, and separate from the system it protects.

Local snapshot storage is useful for development, testing, and fast rollback. Production recovery usually needs external storage such as object storage, cloud blob storage, or a managed backup repository.

For vector databases, storage must preserve enough data to restore objects, vectors, metadata, schema, indexes, tenants, and access-control state.

Why Snapshot Storage Matters

A snapshot is only useful if it survives the failure you are trying to recover from.

If the snapshot is stored on the same failed disk, node, or cluster, it may disappear with production data.

Snapshot storage design is therefore part of disaster recovery, not just an implementation detail.

Local Filesystem Storage

Local filesystem storage saves snapshots or backups to a path on a server, container volume, or attached disk.

It is simple and useful for development, testing, and single-node experiments.

It is usually not enough for production because the backup may be lost if the host or disk fails.

Shared Filesystem Storage

Shared filesystem storage keeps snapshots on a mounted network filesystem or shared storage service.

It can be useful when several nodes need access to the same backup location.

Its reliability depends on the durability, permissions, throughput, and failure behavior of the shared storage platform.

Object Storage

Object storage is a common production choice for snapshot and backup storage.

Examples include S3-compatible storage, Google Cloud Storage, Azure Blob Storage, and private object storage systems.

Object storage is attractive because it is durable, scalable, API-accessible, and often supports lifecycle policies, versioning, encryption, and cross-region replication.

Cloud Blob Storage

Cloud blob storage is object storage provided by a cloud platform.

It is commonly used for database backups because it decouples the backup from the database instance.

If the database cluster becomes unreachable, the snapshot data can still be available from the storage provider.

S3-Compatible Storage

S3-compatible storage is widely used for production backup targets.

It may be hosted by a cloud provider, a managed storage vendor, or an internal object storage system.

Important settings include bucket name, path prefix, endpoint, region, SSL usage, credentials, and permissions.

GCS Storage

Google Cloud Storage is another common backend for snapshot and backup data.

It can store database backup objects, manifests, and exported point-in-time data.

As with any cloud storage target, access credentials and bucket permissions should be managed carefully.

Azure Blob Storage

Azure Blob Storage can store snapshot and backup data in containers.

Configuration usually includes container name, optional path prefix, account credentials, connection strings, and upload settings.

It is suitable for production when the database environment is already on Azure or when cross-cloud recovery requires it.

Backup Repository Storage

Some organizations use a dedicated backup repository rather than raw object storage.

The repository may add cataloging, retention enforcement, immutability, encryption, compression, deduplication, and audit logs.

This can make recovery operations easier to govern.

Internal vs External Storage

Internal snapshot storage lives inside the database environment.

External snapshot storage lives outside the database environment.

External storage is preferred for production recovery because the snapshot remains available even if the database node, volume, or cluster is unavailable.

Failure Domains

A failure domain is the boundary of what can fail together.

Storing snapshots on the same node protects against some software mistakes but not node loss. Storing them in the same region protects against node loss but not regional outage.

Snapshot storage should match the failure scenarios the recovery plan must survive.

Cross-Region Storage

Cross-region storage keeps a copy of snapshot data in another region.

This improves resilience against regional outages, cloud account incidents, or storage failures.

The trade-offs are higher cost, replication delay, and more complex restore planning.

Retention Policies

Retention policies define how long snapshots are kept.

Short retention reduces storage cost but limits recovery from delayed corruption or accidental deletion.

Many systems keep frequent recent snapshots and less frequent older snapshots.

Lifecycle Policies

Lifecycle policies automate storage transitions and deletion.

For example, recent snapshots may stay in fast storage while older snapshots move to cheaper archival storage.

Lifecycle rules should be aligned with restore time requirements, not only cost targets.

Storage Cost

Snapshot storage cost depends on data size, change rate, retention length, compression, replication, and storage tier.

Vector databases can generate large backups because they store objects, vectors, metadata, and indexes.

Incremental backups can reduce cost by storing only changed data after a base backup.

Incremental Snapshot Storage

Incremental snapshot storage depends on a base snapshot or backup.

Unchanged data may be referenced instead of copied again.

Base snapshots and intermediate increments must remain available for as long as any dependent recovery point is needed.

Compression

Compression can reduce snapshot storage size and transfer time.

The trade-off is CPU usage during backup creation and restore.

Compression settings should be tested against realistic database size and recovery targets.

Network Bandwidth

Large snapshots require enough network bandwidth to upload and restore within the required time.

Backup creation may be successful but still too slow for operational needs.

Restore tests should measure actual transfer and load time.

Restore Speed

Snapshot storage affects restore speed.

Fast local or regional storage can restore quickly. Archival or cross-region storage may take longer.

Recovery Time Objective should guide storage tier choice.

Security

Snapshot storage often contains the same sensitive data as production.

It should use encryption, least-privilege access, audit logging, key management, and deletion protection where appropriate.

Backups should not become an easier path to sensitive data.

Immutability

Immutable snapshot storage prevents changes or deletion for a defined period.

This helps protect against accidental deletion, malicious activity, and some ransomware scenarios.

Immutability must be planned carefully because it can also prevent legitimate cleanup before retention expires.

Access Control

Only trusted systems and operators should create, read, restore, or delete snapshot data.

Credentials should be rotated, scoped, and stored securely.

Restore permissions should be controlled because restoring data can expose sensitive historical state.

Monitoring

Snapshot storage should be monitored for failed uploads, missing snapshots, storage growth, replication lag, access errors, retention failures, and restore failures.

Monitoring should include both backup job status and storage backend health.

A snapshot that silently fails to upload is not a recovery point.

Vector Database Considerations

Vector database snapshots may include large vectors, object data, metadata, index files, and manifests.

Indexes can make backups larger, but they may reduce restore time by avoiding expensive rebuilds.

Storage plans should account for collection size, vector dimensions, index type, tenant count, and metadata volume.

What to Store

A useful snapshot storage plan should preserve:

- backup or snapshot data
- restore manifests
- schema or collection definitions
- object and vector data
- metadata and permissions
- index files where supported
- base backups for incremental chains
- logs needed for point-in-time recovery

Common Mistakes

Common mistakes include:

- storing snapshots only on the production node
- using local filesystem storage for production disaster recovery
- deleting base snapshots needed by incremental backups
- choosing cheap storage that restores too slowly
- not encrypting backup data
- giving broad access to backup buckets
- not monitoring storage growth or failed uploads
- never testing restore from the actual storage backend

Practical Checklist

Before relying on snapshot storage, confirm:

- where snapshots are stored
- which failure domains the storage survives
- how long snapshots are retained
- whether storage is encrypted
- who can read and delete snapshot data
- whether lifecycle policies match recovery needs
- whether incremental chains are intact
- how long restore takes from that storage tier

- whether restore has been tested recently

Summary

Snapshot storage determines whether a snapshot is merely convenient or actually useful for recovery.

Production systems should store recovery snapshots outside the primary failure domain, protect them with strong access controls, manage retention, monitor jobs, and test restores.

For vector databases, storage planning must account for objects, vectors, metadata, schema, indexes, tenants, and the size and speed requirements of real restores.